

# 异构存储感知的 Ceph 存储系统数据放置方法

刘 飞<sup>1</sup> 蒋德钧<sup>1</sup> 张 欢<sup>1</sup> 陈 静<sup>2,3</sup> 王 筠<sup>2</sup> 熊 劲<sup>1</sup>

(中国科学院计算技术研究所 北京 100190)<sup>1</sup>

(山东省计算中心(国家超级计算济南中心)山东省计算机网络重点实验室 济南 250101)<sup>2</sup>

(山东科技大学计算机科学与工程学院 青岛 266590)<sup>3</sup>

**摘 要** Ceph 分布式存储系统正成为广泛使用的开源云环境存储解决方案。异构存储如果应用有效的数据管理策略,则能够在保持低成本的同时提供大容量和高性能存储。在 Ceph 中使用异构存储设备不能有效发挥异构存储设备的性能,由于数据的多个副本可以存放到不同的存储介质中,因此不同的副本组合的性能和成本都不一样。针对 Ceph 提出一种面向异构存储的数据放置方法,通过划分多种不同的副本组合,根据数据热度和读写比例将不同的数据放到不同的副本组合上,在提升系统性能的同时有效地控制了系统容量成本。

**关键词** 异构存储,数据放置,副本,Ceph

**中图法分类号** TP311.5 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.06.003

## Heterogeneous Storage Aware Data Placement of Ceph Storage System

LIU Fei<sup>1</sup> JIANG De-jun<sup>1</sup> ZHANG Huan<sup>1</sup> CHEN Jing<sup>2,3</sup> WANG Jun<sup>2</sup> XIONG Jin<sup>1</sup>

(Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China)<sup>1</sup>

(Shandong Provincial Key Laboratory of Computer Networks, Shandong Computer Science Center

(National Supercomputer Center in Jinan), Jinan 250101, China)<sup>2</sup>

(College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China)<sup>3</sup>

**Abstract** Ceph distributed storage system is becoming a widely used open source cloud storage solution. Heterogeneous storage can provide large capacity and high performance storage while maintaining low cost if it uses an effective data management strategy. Using heterogeneous storage devices in Ceph currently cannot effectively exploit the performance of heterogeneous storage devices. Since multiple replicas of the data can be stored in different storage media, the performance and cost of different device combinations for multiple replicas are not the same. In this paper, a data placement method was proposed for heterogeneous storage based on Ceph. The method puts different data on different replica combination based on the access intensity and read/write ratio, which can effectively improve the system performance while controlling the cost of the system.

**Keywords** Heterogeneous storage, Data placement, Replicas, Ceph

## 1 引言

云计算服务的推广和广泛应用使得数据存储压力持续增长。云时代的存储系统面对如此大的用户量,必须具备高性能和大容量等特点。由于 HDD 本身的性能限制,传统的基于 HDD 的存储系统难以适用于实时性强的应用场景。相对于 HDD, SSD 虽然拥有出众的 I/O 性能,但是其单位容量的价格较高,如果在云环境存储系统中使用 SSD 来代替 HDD,会使得存储系统的成本过高;而基于 HDD 和 SSD 的异构存储如果应用有效的数据管理策略,则能够在保持低成本的同

时提供大容量和高性能存储。

Ceph<sup>[1-3]</sup> 分布式存储系统被广泛用作云环境开源存储解决方案。Ceph 不感知异构存储,会把数据均匀地分散到各个存储设备上。若在 Ceph 中使用异构存储设备,现有的数据放置策略不能有效发挥异构存储设备的性能,原因如下:

- 1) 数据的多个副本可以存放到不同的存储介质中;
- 2) 不同副本组合的性能和成本都不一样;
- 3) 现有的数据放置策略没有充分利用多副本的各种组合方式。

本文的主要工作内容和贡献如下:

到稿日期:2016-11-11 返修日期:2017-01-04 本文受山东省计算机网络重点实验室开放课题基金(SDKLCN-2013-01),山东省自然科学基金项目(ZR2016FM41),国家青年自然科学基金项目(61502448)资助。

刘 飞(1991—),男,硕士,主要研究方向为分布式存储,E-mail:liufei01@ict.ac.cn;蒋德钧(1982—),男,博士,副研究员,主要研究方向为分布式存储、云计算;张 欢(1981—),男,硕士,主要研究方向为分布式存储;陈 静(1978—),女,博士生,副研究员,主要研究方向为云计算、软件测试;王 筠(1976—),女,硕士,助理研究员,主要研究方向为软件工程、云计算;熊 劲(1968—),女,博士,研究员,主要研究方向为分布式存储、大数据存储。

1) 针对分布式存储系统中的多副本场景, 提出了一种异构存储副本组合方式;

2) 针对本文所研究的异构存储, 基于 Ceph 现有的数据映射方法, 设计了一种区分异构存储设备的数据映射策略;

3) 设计了一种数据冷热统计方法, 该方法是基于多版本布隆过滤器<sup>[4]</sup>的, 并且能区分读写请求;

4) 基于副本组合方式提出一种数据迁移机制, 从数据的读写比例和热度两个维度进行考虑, 选择合适的副本组合来存放数据。

本文第 2 节介绍关键设计问题; 第 3 节介绍 Ceph 异构存储系统的实现; 第 4 节对所提数据放置方法进行评测; 第 5 节介绍相关的工作; 最后总结全文。

## 2 关键设计问题

本文的异构存储数据放置策略所要解决的关键问题包括: 副本组合方式、数据映射、数据热点识别和数据迁移。

### 2.1 副本组合方式

在分布式存储系统中, 由于数据的多个副本可以存放在不同存储介质上, 而不同的副本组合具有不同的特性, 因此数据多副本如何组合是首先要解决的问题。以数据三副本为例 (三副本是最常见的场景), 其副本有如下 4 种组合方式: 3HDD, 2SSD+1HDD, 1SSD+2HDD 和 3HDD。

如图 1 所示, 在数据强一致性的场景下, Ceph 读数据时只需要从一个副本上读取, 所以三副本只需要有一个副本放到 SSD 上就能显著提升读性能; 而写数据需要写完三副本, 三副本有一个 HDD 就会拖慢整个写的过程, 所以需要三副本全放到 SSD 上才能提升写的性能。

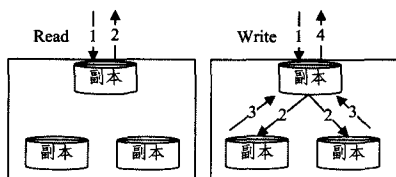


图 1 Ceph 数据的读写模式

由于 2SSD+1HDD 相对于 1SSD+2HDD 不仅在性能上没有提升, 而且使用了较多的 SSD, 因此将副本组合分为如下 3 类。

- 1) 3SSD: 读写性能都比较好;
- 2) 1SSD+2HDD: 读性能好, 写性能差;
- 3) 3HDD: 读写性能都比较差。

### 2.2 数据映射

异构存储系统中由于存在多种设备类型, 并且数据不是固定存放在某一种设备类型上的, 随着应用负载的变化, 数据在多种设备之间动态迁移, 因此需维护一些元数据信息, 这些元数据用来记录数据和设备的对应关系。

本文所讨论的数据映射是针对异构存储引入的额外元数据信息所采用的映射方法。2.1 节把副本组合分为 3 类, 数据映射需要解决的即是数据存放在哪一类副本组合中的问题。

大部分异构存储系统中都用一个映射表 (Mapping Ta-

ble) 来维护数据到不同存储设备的映射关系<sup>[5,15,18]</sup>, 然而 Ceph 没有这种统一的数据映射表, 它利用集群状态图 cluster map 和 CRUSH 算法<sup>[2]</sup>来计算数据所在 OSD。可以通过添加一个映射表来记录数据所在的副本组合, 从而结合映射表和原有的数据映射算法就可以计算出正确的数据位置, 但是这种方法不仅存在额外的查表开销, 而且映射表的大小会随着数据量的增大而线性增长。

由上述可知, 采用映射表的方法存在很多问题, 扩展性不强, 所以本文采用逐类副本组合查找的方法。

### 2.3 热点数据识别

本系统利用多版本布隆过滤器<sup>[4]</sup>的方法来统计对象的历史访问信息, 多版本布隆过滤器的优点如下:

- 1) 内存占用少;
- 2) 计算复杂度低;
- 3) 有效地抓住了时间维度。

如图 2 所示, 假设使用  $n$  版本布隆过滤器, 那么总共存在  $n-1$  个历史版本布隆过滤器和 1 个当前版本布隆过滤器, 不同版本的布隆过滤器记录了不同时间段的对象访问信息。对象当前的访问信息记录在当前版本的布隆过滤器中, 经过一段时间或者记录一定数目的请求后就会把当前版本插入到历史版本列表中并产生一个新的布隆过滤器, 如果历史版本数超出了系统所设置的最大版本个数限制, 就把最早的历史版本删除。

| Current | History 1 | History n-1 | Deleted |
|---------|-----------|-------------|---------|
| 0 1     | 0 0       | 0 0         | 0 0     |
| 0 0     | 0 0       | 0 1         | 0 0     |
| 1 1     | 1 1       | 1 1         | 1 1     |
| 0 0     | 0 0       | 0 0         | 0 0     |
| 1 1     | 1 1       | 1 1         | 1 1     |
| 0 0     | 0 1       | 0 0         | 0 0     |
| 0 1     | 0 0       | 0 1         | 0 1     |
| 0 0     | 0 0       | 0 0         | 0 0     |

图 2 多版本布隆过滤器

图 2 中布隆过滤器的每个存储单元包含两位, 都初始化为 0, 其中一位用来区分读写请求, 0 表示读, 1 表示写; 另一位用来表示是否访问过, 0 表示没有访问过, 1 表示访问过。

每个 I/O 请求都会把所请求的对象插入到当前版本的布隆过滤器中, 因为当前版本的布隆过滤器会存在一段时间, 在这段时间内可能会有多次对同一个对象的访问, 而且可能是读写交叉访问, 而读写标志位只有一位。读写标志位的修改应遵循以下原则:

- 1) 对于读请求, 在插入的时候不修改读写标志位;
- 2) 对于写请求, 在插入的时候会把读写标志位设为 1。

也就是说, 对于同一个对象的访问信息, 写请求会覆盖读请求, 而读请求不会覆盖写请求。这是因为在数据迁移时写频繁的数据所迁移的目标是读写性能都好的副本组合, 而读频繁的数据所迁移的目标是读性能好而写性能不好的副本组合。

利用上述方法统计对象历史访问信息后, 通过统计对象在所有布隆过滤器中出现的次数来判断对象访问的冷热情况, 并结合读写标志位进一步得出读写的情况。



## 2.4 数据迁移

不同的副本组合适合存放不同的数据,所以需要根据数据的历史访问特征将数据迁移到合适的副本组合中。

数据迁移存在如下3种情况。

(1)由3HDD或者1SSD+2HDD迁入3SSD

迁移数据:写访问达到阈值的对象,例如在对象的当前 Bloom Filter 中出现,并且总共出现的 Bloom Filter 数超过一半,读写标志位为1的超过一半。

(2)由3HDD或者3SDD迁入1SSD+2HDD

迁移数据:写访问达到阈值的对象。

(3)由3SDD或者1SSD+2HDD迁入3HDD

迁移数据:读写访问低于阈值的对象。

## 3 Ceph 异构存储系统的实现

本节介绍所提出的数据放置方法在 Ceph 原型系统中的实现。

Ceph 原型系统的数据放置如图3所示。对象通过 hash 算法映射到放置组 (Placement Group, PG) 中,然后通过 CRUSH 算法将 PG 映射到一组 OSD(Object Storage Device) 中,最后,对象就找到了属于它的一组 OSD。

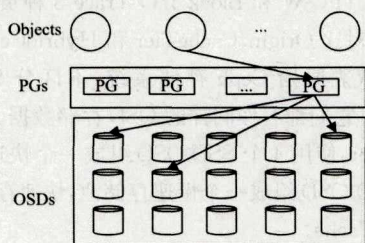


图3 Ceph的数据放置

Ceph 异构存储的系统架构如图4所示。将副本组合进行分类,即 PG 分类(Ceph 中副本单元为 PG),将 PG 分类之后改变了原有的数据映射流程。对象首先映射到某一类 PG,某一类 PG 映射到特定组合的 OSD,I/O 请求发送到 OSD 之后会经过 OSD 端的数据热点识别模块,经过一段时间的历史访问信息统计之后,数据迁移模块就会根据数据迁移策略将数据迁移到合适的副本组合中。

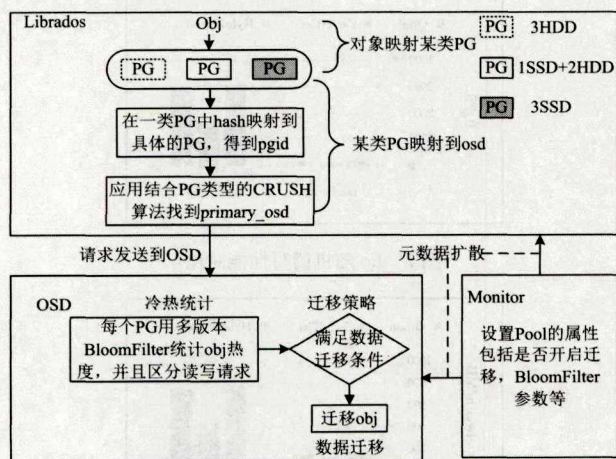


图4 Ceph异构存储系统架构

### 3.1 数据映射

数据映射过程中 I/O 请求会依次查找3类 PG。对于如

何查找对象是否存在于某一类 PG 上,我们添加了 check\_object\_exist 函数。该函数首先会查看对象文件的状态,如果返回成功则说明对象在该类 PG 上,如果返回失败则说明对象不在该类 PG 上。

对象映射到某一类 PG 后,会在该类 PG 中被哈希映射到具体的 PG 上,然后通过修改后的 CRUSH 算法来计算出符合条件的 OSD 集合。在 Ceph 原有的 CRUSH 算法上添加对 PG 类别的识别,通过不同的 PG 类别选择预先编写好的 CRUSH 规则就可以计算出与 PG 类别对应的 OSD。

### 3.2 数据热点识别

数据热点识别是在 OSD 模块 ReplicatedPG::do\_op 函数中进行的,所有对象的读写操作都会经过该函数进行预处理。在 OSD 启动时如果设置了开启数据冷热统计的相关参数,那么系统就会初始化 BloomFilter 数据结构和用来存放历史版本 Bloom Filter 的 List。BloomFilter 的 bittable 使用 char 指针类型,并通过为 char 指针分配一块内存空间来存放对象的访问信息,这块内存空间中每两个 bit 作为一组来代表哈希的一个位置,两个 bit 的其中一位存放读写信息,另外一位存放访问信息。

判断数据冷热时,需首先查看存放访问信息的 bit,若对象在当前布隆过滤器中出现并且在历史版本的布隆过滤器中出现的次数超过历史版本数的一半,则认为该数据是热点数据。查看当前的布隆过滤器是为了确定对象在最近被访问过,查看历史版本布隆过滤器是为了确定该对象属于在之前就频繁访问的对象;然后查看读写标志位,若读写标志位超过一半为1,则表示该对象为写频繁的对象,否则为读频繁的对象。

### 3.3 数据迁移

在达到迁移间隔条件,确定迁移数据集和迁移目标后,就需要把数据迁移到目标副本组合上。假设对象从 C 类 PG 迁移到 B 类 PG,会经历如下步骤。

1)把要迁移的对象加入到迁移集合中,该迁移集合表示所有正在迁移的对象的集合,目的在于防止在对象迁移过程中迁移模块又选中该对象进行迁移,造成重复迁移,选择迁移数据集时会首先查找该对象是否存在于正在迁移的集合中,如果存在则不会迁移该对象;

- 2)在迁移目标 PG 上阻塞对于迁移对象的请求;
- 3)在 C 类 PG 上读出对象;
- 4)将其写到目标 PG 上;
- 5)删除 C 类 PG 上的对象;
- 6)解除目标 PG 对迁移对象访问请求的阻塞;
- 7)将迁移对象从迁移集合中删除。

## 4 实验与评价

本节对所提出的异构存储数据放置方法进行评测。4.1 节介绍测试平台、负载以及评测系统;4.2 节对数据映射所产生的开销进行了评测;4.3 节对系统的整体性能进行评测,并进行了分析;最后进行总结。

### 4.1 测试平台、负载以及评测系统

测试平台如图5所示,使用两台服务器搭建存储集群,每



台服务器有 12 个物理核,64GB 内存,9 块 HDD 和 1 块 SSD,其中一块 HDD 作为系统盘,运行于 CentOS 7 操作系统。存储集群包含 20 个 OSD,其中有 16 个 HDD OSD 和 4 个 SSD OSD,每台服务器上 8 块 HDD 做成 8 个 HDD OSD,1 块 SSD 分 2 个区做成 2 个 SSD OSD。使用虚拟机作为客户端来访问 Ceph 集群,通过虚拟机挂载存储集群提供的块存储设备并对块存储设备进行读写来测试存储集群的性能,这也是 Ceph 的典型应用场景。

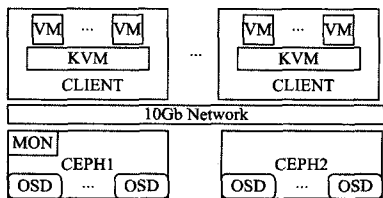


图 5 测试平台

采用的负载如下:

1) fio<sup>[6]</sup>。fio 是一个强大的 I/O 性能测试工具,本文采用的测试参数如表 1 所列。

表 1 fio 参数

| 参数       | 顺序     | 随机     |
|----------|--------|--------|
| ioengine | libaio | libaio |
| direct   | 1      | 1      |
| bs       | 64k    | 4k     |
| iodepth  | 64     | 8      |

2) TPC-W<sup>[6]</sup>。TPC-W 是一个事务型 Web 基准程序,包含 BrowsingMix, ShoppingMix 和 OrderingMix 3 种访问模式,读写比率分别为 95%:5%,80%:20%和 50%:50%。

3) Block I/O Traces<sup>[8]</sup>;数据中心服务器 I/O Traces。

本文的性能评测主要是通过 Ceph 原型系统以及 Ceph Cache Tier 进行对比,所有系统都采用数据三副本模式。

1) Ceph 原型系统

开源的 Ceph 存储系统,将 SSD 作为与 HDD 一样的 OSD,没有添加智能的异构存储数据管理策略。

2) Ceph Cache Tier

在 Ceph 基础存储集群上建立两个存储池,一个是用 SSD OSD 组成的快速存储池,另一个是用 HDD OSD 组成的慢速存储池,快速存储池作为慢速存储池的 Cache。

## 4.2 映射开销测试

本文所提异构存储数据映射方法是逐个查找不同类别的副本组合的方式,可能会有一定的查找转发开销。

为了测试查找转发开销,本文所设计的 Ceph 异构存储系统在测试过程中会关掉数据迁移,所有的数据都需要查找前两类 PG,并且都会由最后一类 PG 服务请求,这也是最坏的数据访问情况。系统配置 3HDD 作为最后一类 PG,即数据请求都会查找 3SSD 和 1SSD+2HDD 类型的 PG 上是否存在请求的数据,因为没有数据迁移,所以 3SSD 和 1SSD+2HDD 类型的 PG 上没有请求的数据,最后,请求由 3HDD 类别的 PG 提供服务。

图 6 示出了 fio 随机读在不同虚拟机个数(客户端个数)的情况下的性能,图中 Origin 代表没有添加异构存储数据管

理机制的 Ceph 存储系统,不用进行 PG 的查找,直接读写 16 个 HDD OSD 的性能;PG 查找代表在查找两类 PG 后没有找到请求的数据,随后读写 16 个 HDD OSD 的性能。从图 6 中可以看出,Origin 和 PG 查找的性能基本一样,最多相差 6%,随着虚拟机个数增多,并发访问量也增大,过大的并发访问所带来的资源竞争等使得系统的性能不仅没有上升反而有所下降,并发访问量的增大也没有增加 PG 查找开销,而是进一步减小了开销。由于高并发访问产生的 I/O 队列的阻塞将本来就很小的查找转发开销基本掩盖,因此其性能基本与 Origin 一样。fio 其他访问模式和图中表现出的结果也一样。

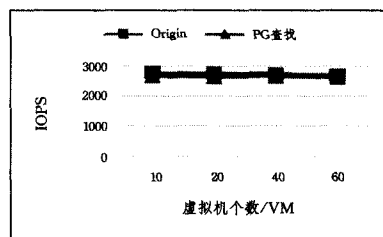


图 6 PG 查找-随机读

## 4.3 性能评测

采用 fio, TPC-W 和 Block I/O Trace 3 种负载对性能进行评测,分别对比 Origin, CacheTier 和 HybridCeph 的性能。

Origin:代表原始 Ceph 存储系统,不区分 SSD OSD 和 HDD OSD,将它们都看作同样的 OSD 存储数据。

CacheTier:使用 4 个 SSD OSD 组成一个快速存储池,另外 16 个 HDD OSD 组成一个慢速存储池,快速存储池作为慢速存储池的 Cache。

HybridCeph:代表本文设计的异构存储系统,将副本组合分类,把读访问频繁的数据和写访问频繁的数据迁移到不同的副本组合上。

(1) fio 测试

通过虚拟机运行 fio 生成顺序读写和随机读写请求来对存储系统进行测试。

图 7 和图 8 分别示出了 fio 随机读写和顺序读写的性能对比情况。

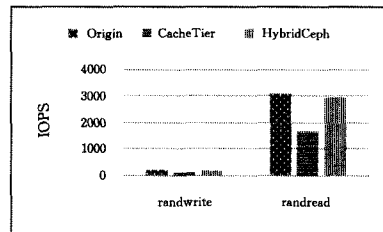


图 7 fio 随机读写性能对比

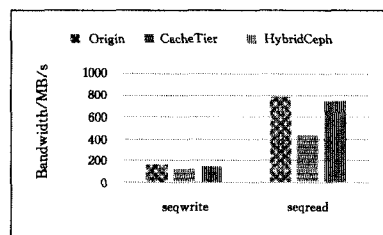


图 8 fio 顺序读写性能对比

从图7、图8中可以看到,CacheTier的性能比Origin的性能差21%~46%,这是因为 fio 生成的数据访问比较均匀,没有热点数据,cache 命中率很低。CacheTier 在 cache 不命中的情况下,数据请求会发送到慢速存储池,同时在后台把访问的对象从慢速存储池 promote 到快速存储池,因为 cache 命中率很低,所以基本上所有的数据请求都是由慢速存储池服务,并且慢速存储池的对象不断地 promote 到快速存储池,快速存储池的对象也会不断地 demote 到慢速存储池,频繁的数据迁移不仅没有提升系统性能,反而对系统性能造成了很大的影响。HybridCeph 在达到一定的时间间隔和请求数目后才会检查是否有需要迁移的数据,只有达到数据访问阈值才会进行数据迁移,所以在 fio 这种没有数据局部性的负载下,数据迁移没有那么频繁,HybridCeph 的性能与 Origin 相差 5%~8%。

## (2) TPC-W 测试

图9所示为 TPC-W 测试的结果,纵坐标是 WIPS(Web Interaction Per Second),WIPS 是 TPC-W 性能的主要衡量指标,横坐标是 3 种访问模式。从图中可以看到,CacheTier 相对于 Origin 有 40%~54% 的性能提升,HybridCeph 和 CacheTier 的性能相差 4%~7%。

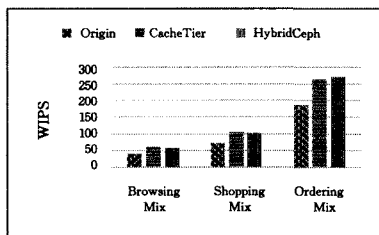


图9 TPC-W 性能测试

虽然 HybridCeph 相对于 CacheTier 没有明显的性能优势,但是由于 HybridCeph 将多副本组合分为 3 类: 3SSD, 1SSD+2HDD 和 3HDD, 相对于 CacheTier 把数据 promote 到 3SSD 上 HybridCeph 把读访问频繁的数据迁移到 1SSD+2HDD 上不仅没有数据冗余,而且显著节省了 SSD 空间。

图10所示为 BrowsingMix 访问模式下 SSD 使用量(SSD 上写入的总数据量)的统计。Browsing Mix 访问模式中基本都是读请求,所以 HybridCeph 基本都把数据迁移到 1SSD+2HDD 多副本组合上,并且由于 HybridCeph 迁移没有 CacheTier 频繁,因此 HybridCeph 的 SSD 使用量少于 CacheTier 的 SSD 使用量的 1/3。从 Browsing Mix, Shopping Mix 到 Ordering Mix, 由于写的比例逐渐增大,因此迁移到 3SSD 上的数据会变多,相对 CacheTier, HybridCeph 的 SSD 使用量比例也逐渐增大,HybridCeph 的 SSD 使用量分别是 CacheTier 的 42% 和 60%。

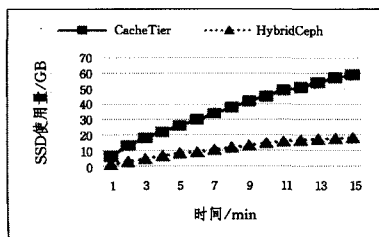


图10 Browsing Mix 访问模式下的 SSD 使用量

## (3) Block I/O Traces 测试

图11所示为数据中心服务器 I/O traces 测试的结果。从图中可以看出,CacheTier 和 HybridCeph 的 trace 回放时间都比 Origin 的 trace 回放时间短。HybridCeph 的 trace 回放时间相对于 Origin 最大缩短了 46%, 最小缩短了 41%; CacheTier 的 trace 回放时间相对于 Origin 最大缩短了 64%, 最小缩短了 32%。在运行 prn\_0 和 radiussql 时, CacheTier 的性能比 HybridCeph 高 30%; 而在运行 adsds 和 webmail 时, HybridCeph 性能比 CacheTier 高 20%。adsds 和 webmail 的数据访问模式属于大量均匀访问(大部分的数据块重复访问次数较多,但是重复访问不是集中在一起,而是比较均匀地分散到整个数据访问过程中), 而 prn\_0 和 radiussql 的数据访问模式属于大量集中访问(大部分的数据块重复访问次数较多并且重复访问比较集中)。由于 CacheTier 不断有数据 promote 到快速存储池和 demote 到慢速存储池, 大量均匀访问的数据在一段时间没有被访问后可能会被 demote 到慢速存储池, 当再次访问时还需从慢速存储池读取数据, 因此性能不好; 而 HybridCeph 能够准确识别出大量均匀访问的数据并迁移到性能好的多副本组合。对于大量集中式访问, HybridCeph 由于存在一定的迁移时间间隔, 因此并不一定能及时把数据迁移到 SSD 上, 因此其性能比 CacheTier 差。由于读访问频繁的数据会迁移到 1SSD+2HDD 多副本组合上, 因此 HybridCeph 的 SSD 使用量比较少, 相对于 CacheTier 少 18%~65%。

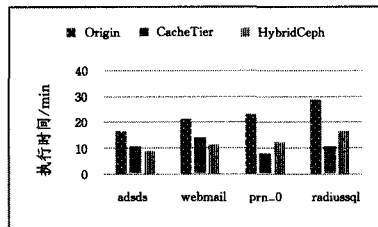


图11 Block I/O traces 性能测试

## 4.4 小结

测试结果表明: 1) 系统的映射开销较小, 可扩展性较强, I/O 并发量增大并不会增加映射开销; 2) 在 fio 这种没有数据局部性的负载下系统性能与原有的 Ceph 存储系统性能相近, 并且相对于 CacheTier 的性能提升了 18%~45%, 在 TPC-W 以及数据中心服务器 trace 这样的具有冷热数据的负载下系统性能相对于原有的 Ceph 存储系统提升了 40%~50%; 3) 系统 SSD 使用量相对于 CacheTier 减少了 18%~65%, 而较少的 SSD 使用量可以降低系统成本。

## 5 相关工作

基于 HDD 和 SSD 的异构存储从组织结构上主要分为两类: 一类是 SSD 作为 HDD 的 Cache, 另一类是 SSD 与 HDD 作为同层的持久化存储。

SSD 作为 Cache 的工作包括 FlashCache<sup>[9]</sup>, Azor<sup>[10]</sup>, SmartSaver<sup>[11]</sup>, Interposing<sup>[12]</sup> 等。FlashCache 是 Facebook 的一个开源项目, 它是在 VFS 和设备驱动之间的 Device Mapper 上实现的一个内核模块, 通过对磁盘数据块的缓存来提升存

储性能。SSD 作为 Cache 的主要缺点是数据冗余、数据一致性维护和 SSD 能量损耗过快。

SSD 作为持久化存储的工作包括 Hybrid Store<sup>[13]</sup>, Hybrid Aggregates<sup>[14]</sup>, Hystor<sup>[15]</sup>, hStorage-DB<sup>[16]</sup>, SSD-HDD-hybrid<sup>[17]</sup>, HRO<sup>[18]</sup>, Hybrid Hbase<sup>[19]</sup>, hatS<sup>[20]</sup>等,本文所研究的异构存储属于 SSD 与 HDD 同层的组织结构。

现有的数据放置和迁移总结如表 2 所列,可以分为数据属性、副本、数据冷热和优先级 4 类。

表 2 数据放置分类

|             | SSD             | HDD     |
|-------------|-----------------|---------|
| 数据属性        | 元数据、日志等         | 数据      |
| 副本          | 单个副本            | 其他副本    |
| 数据冷热、I/O 粒度 | 热数据 & 小粒度随机 I/O | 其他数据    |
| 优先级         | 优先级高的请求         | 优先级低的请求 |

与传统的本地异构存储系统不同,在使用多副本的分布式存储系统中,多个副本可以不存放于单一介质中,并且不同的副本组合的特性也不一样。现有的异构存储数据放置没有充分利用多副本的各种组合方式。

本本文的工作最接近的是 hatS<sup>[20]</sup>,其在 HDFS 中使用了 3 种不同特征的存储设备(Ramdisk, SSD, HDD),每种存储设备划分为一个 Tier,数据的三副本存放在 3 个不同的 Tier 和 Node 上,在读数据时根据 Node 的负载、存储设备的利用率等来确定从哪一个副本读取数据。该方法虽然提升了读的性能,但是在数据强一致性的情况下,数据写完三副本才算写成功, HDD 会拖慢写的速度,没有优化写的性能。

**结束语** 本文以 Ceph 为原型研究了分布式存储系统中异构存储数据的管理方法。其提出了一种面向异构存储的副本组合方式,并基于该组合方式提出了一种数据放置方法,根据不同的数据访问模式选择合适的副本组合,不仅提升了系统整体的性能,而且还有效控制了成本。

## 参考文献

- [1] WEIL S A, BRANDT S A, MILLER E L, et al. Ceph: A scalable, high-performance distributed file system[C]// Proceedings of the 7th Symposium on Operating Systems Design and Implementation, USENIX Association, 2006: 307-320.
- [2] WEIL S A, BRANDT S A, MILLER E L, et al. CRUSH: Controlled, scalable, decentralized placement of replicated data[C]// Proceedings of the 2006 ACM/IEEE Conference on Supercomputing. ACM, 2006: 122.
- [3] WEIL S A, LEUNG A W, BRANDT S A, et al. Rados: a scalable, reliable storage service for petabyte-scale storage clusters [C]// Proceedings of the 2nd International Workshop on Petascale Data Storage: Held in conjunction with Supercomputing'07. ACM, 2007: 35-44.
- [4] PARK D, DU D H C. Hot data identification for flash-based storage systems using multiple bloom filters[C]// 2011 IEEE 27th Symposium on Mass Storage Systems and Technologies (MSST). IEEE, 2011: 1-11.
- [5] CAI Y J, KANG C K, WU C H. A virtual storage environment

for SSDs and HDDs in Xen hypervisor[C]// ACM SIGBED. 2014: 39-44.

- [6] fio[OL]. <http://freecode.com/projects/fio/>.
- [7] TPC-WHomepage[OL]. <http://www.tpc.org/tpcw/>.
- [8] Block I/O Traces[OL]. <http://iotta.snia.org/traces/>.
- [9] FlashCache[OL]. <https://github.com/facebook/flashcache>.
- [10] KLONATOS Y, MAKATOS T, MARAZAKIS M, et al. Azor: using two-level block selection to improve SSD-based I/O caches [C]// 2011 6th IEEE International Conference on En Networking, Architecture and Storage (NAS). IEEE, 2011: 309-318.
- [11] CHEN F, JIANG S, ZHANG X. SmartSaver: turning flash drive into a disk energy saver for mobile computers[C]// International Symposium on Low Power Electronics and Design, 2006 (ISLPED'06). IEEE, 2006: 412-417.
- [12] KHATIB M G, VAN DER ZWAAG B J, Hartel P H, et al. Interposing flash between disk and DRAM to save energy for streaming workloads[C]// IEEE/ACM/IFIP Workshop on Embedded Systems for Real-Time Multimedia, 2007 (ESTIMedia 2007). IEEE, 2007: 7-12.
- [13] KIM Y, GUPTA A, URGANONKAR B, et al. HybridStore: a cost-efficient, high-performance storage system combining SSDs and HDDs[C]// 2011 IEEE 19th International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS). IEEE, 2011: 227-236.
- [14] STRUNK, JON D. Hybrid Aggregates: Combining SSDs and HDDs in a single storage pool[J]. ACM SIGOPS Operating Systems Review, 2012, 46(3): 50-56.
- [15] CHEN F, KOUFATY D A, ZHANG X D. Hystor: making the best use of solid state drives in high performance storage systems[C]// Proceedings of the International Conference on Supercomputing. ACM, 2011: 22-32.
- [16] LUO T, LEE R, MESNIER M, et al. hStorage-DB: heterogeneity-aware data management to exploit the full capability of hybrid storage systems[J]. Proceedings of the VLDB Endowment, 2012, 5(10): 1076-1087.
- [17] JO H, KWON Y, KIM H, et al. SSD-HDD-hybrid virtual disk in consolidated environments [M] // En Euro-Par 2009-Parallel Processing Workshops. Springer Berlin Heidelberg, 2010: 375-384.
- [18] LIN L, et al. Hot random off-loading: a hybrid storage system with dynamic data migration[C]// 2011 IEEE 19th International Symposium on En Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS). IEEE, 2011: 318-325.
- [19] AWASTHI A, NANDINI A, Bhattacharya A, et al. Hybrid HBase: Leveraging flash SSDs to improve cost per throughput of HBase[C]// Proceedings of the 18th International Conference on Management of Data. Computer Society of India, 2012: 68-79.
- [20] KRISH K R, ANWAR A, BUTT A R. hatS: A heterogeneity-aware tiered storage for Hadoop[C]// 2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid). IEEE, 2014: 502-511.